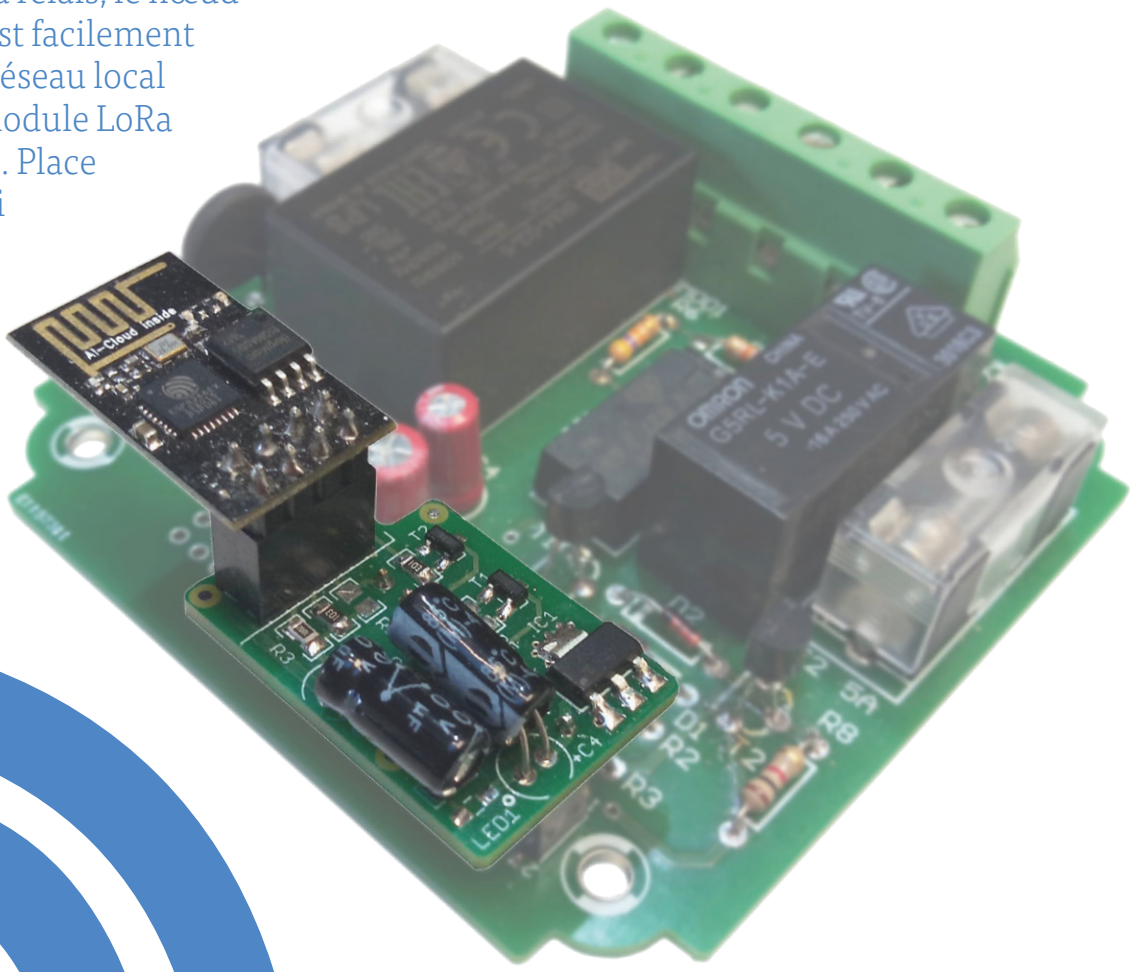
wifi pour le nœud LoRa d'Elektor

Interrupteur intégré dans Home Assistant avec ESPHome

Clemens Valens (Elektor)

Avec sa belle carte à relais, le nœud LoRa d'Elektor [1] est facilement utilisable dans un réseau local en remplaçant le module LoRa par un modèle Wifi. Place à l'interrupteur wifi d'Elektor.



Un interrupteur télécommandé a fait l'objet d'un article dans le magazine *Elektor* de mars-avril 2020 [1]. Il dispose du retour d'état du relais et communique sur le réseau LoRa. Logé dans un boîtier étanche IP66, il est adapté à une utilisation en extérieur. C'est également un projet modulaire avec le relais et le circuit d'alimentation sur une carte et la partie communication LoRa sur une autre.

Il a retenu mon attention, car je cherche un interrupteur extérieur, facile à intégrer à mon système de domotique. Ce dernier repose sur le logiciel Home Assistant et le réseau wifi, mais pas sur LoRa. Je suis sûr qu'on peut y ajouter LoRa, mais je n'ai pas envie de chercher. À la place, j'espère qu'en remplaçant simplement le module LoRa par un module Wifi, on peut y exécuter le *framework* ESPHome, qui fonctionne très bien avec Home Assistant (**fig. 1**).

Description de l'interface

La première chose à faire quand on essaie d'adapter quelque chose est de trouver comment s'y connecter. Le schéma de la carte de l'interrupteur n'est pas très compliqué (**fig. 2**). En haut, nous avons l'alimentation électrique avec un convertisseur CA/CC MOD1 qui fournit 5 V avec quelques éléments de protection et de filtrage.

Au centre, il y a un relais bistable RE1, qui allume et éteint l'alimentation de la charge. Un relais bistable ressemble beaucoup à un interrupteur mécanique, car tout comme lui il conserve son état même après avoir été mis hors tension. Deux signaux permettent de basculer son état : *Set* et *Reset*. Ils sont pilotés par deux MOSFET T1 et T2 commandés par des signaux actifs à l'état haut.

Vous noterez que le relais est protégé par un fusible F2 de 5 A alors que le relais peut commuter jusqu'à 16 A. C'est parce que les pistes du circuit imprimé ne peuvent pas supporter un tel courant, donc le fusible les protège contre la fusion. La carte de l'interrupteur peut supporter des charges allant jusqu'à environ 1 kW.

Dans le coin inférieur gauche, on trouve un optocoupleur IC1 en parallèle avec la charge. Il est activé en même temps que la charge. Sa LED rend son transistor conducteur et la sortie passe à l'état bas. Lorsque la charge n'est pas alimentée, la sortie est à l'état haut. Le signal CA à 50 ou 60 Hz est filtré par C4 et R4 pour maintenir un niveau bien stable.

Le connecteur dans le coin inférieur droit permet d'accéder à tous les signaux nécessaires. S1 est destiné à un bouton-poussoir placé sur l'appareil, pour pouvoir aussi actionner l'interrupteur localement.

Il faut savoir que l'optocoupleur est connecté à une alimentation de 3,3 V, mais il n'y en a pas sur la carte. Cette tension doit être fournie par la carte de commande.

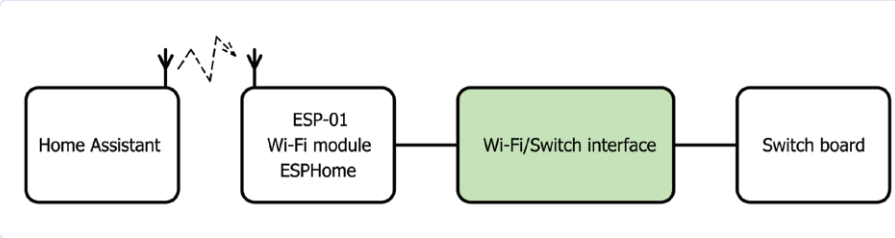


Figure 1. Vue d'ensemble du système. Dans cet article, nous nous intéressons principalement au bloc vert clair.

Le module Wifi ESP-01

Nous pouvons maintenant établir les spécifications de la carte de commande wifi qui pilotera la carte de l'interrupteur :

- Il faut deux entrées numériques, une pour lire la sortie de l'optocoupleur et une pour S1 ;
- Il faut deux sorties numériques pour commander T1 et T2 ;
- Il faut fournir 3,3 V pour l'optocoupleur.

Maintenant, quand vous lisez « quatre ports d'E/S numériques, wifi et 3,3 V », vous pensez bien sûr immédiatement au module ESP-01, car il a exactement quatre ports GPIO, le wifi

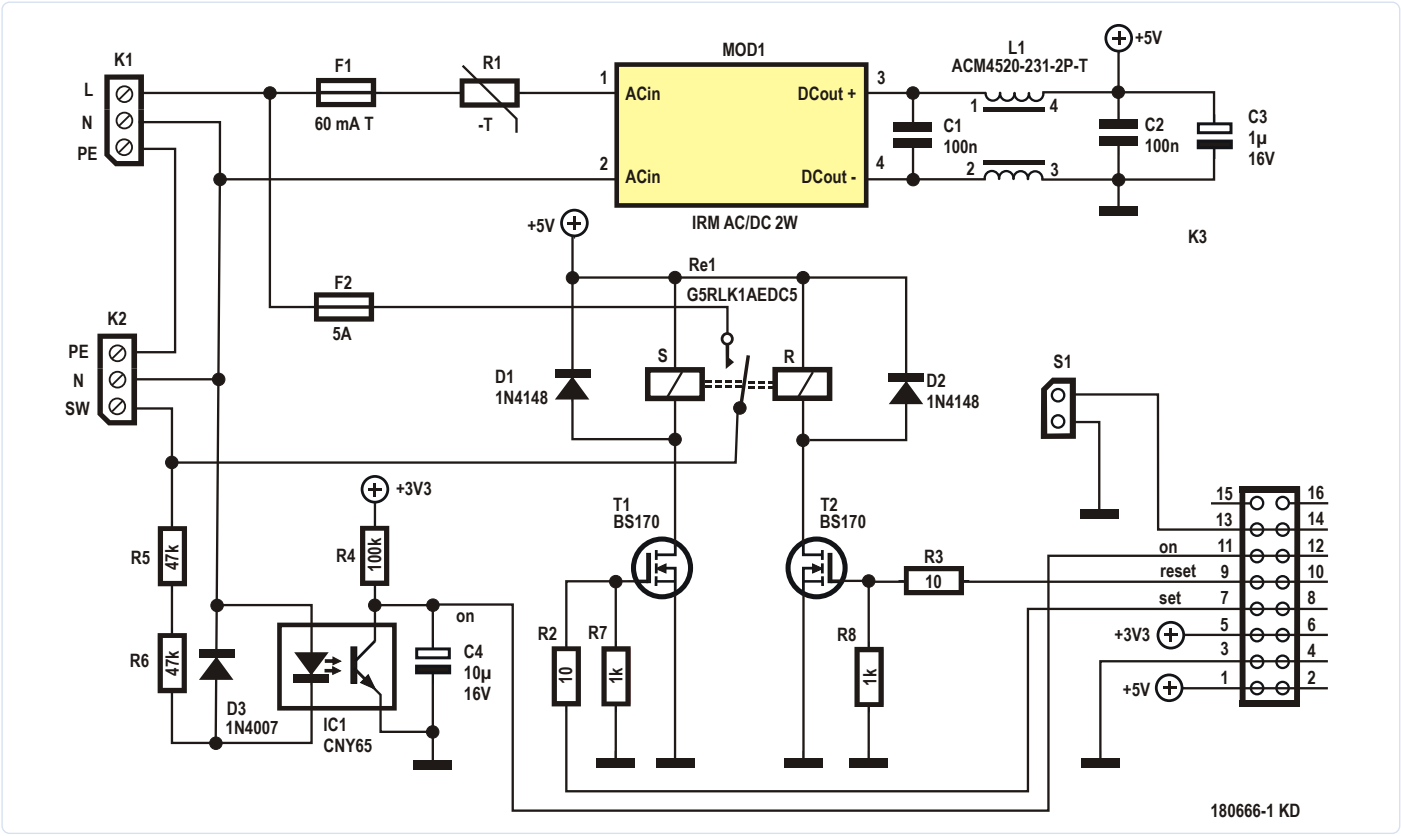


Figure 2. La carte de l'interrupteur du nœud Lora d'Elektor (mars-avril 2020, [1]) est centrée autour d'un relais bistable.

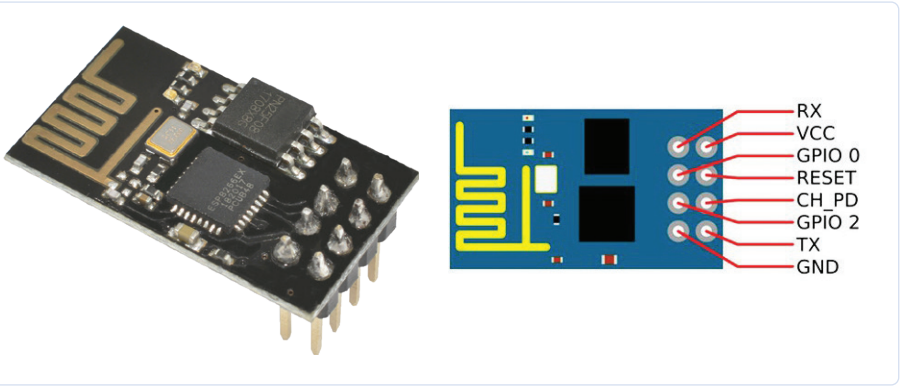


Figure 3. Le module ESP-01, petit et bon marché, est équipé d'un microprocesseur ESP8266EX avec wifi intégré. Il possède quatre ports d'E/S (GPIO 0, GPIO 2, RX & TX) et nécessite une alimentation en 3,3 V (VCC).

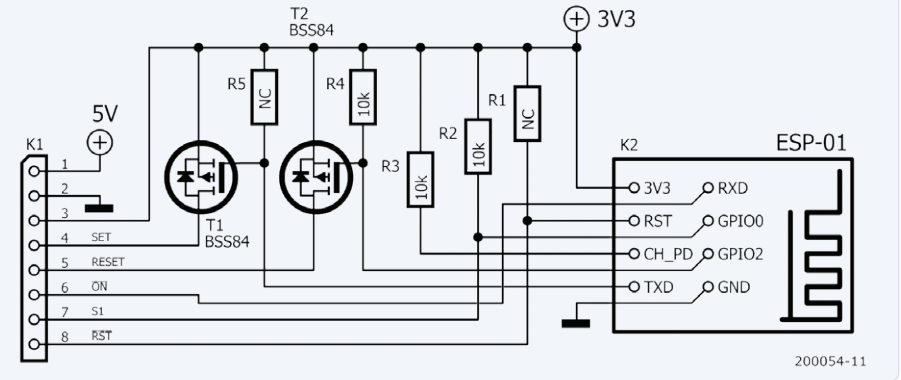


Figure 4. Une façon possible de connecter un module ESP-01 à la carte de l'interrupteur LoRa. Les MOSFET découplent les entrées des sorties.

LISTE DES COMPOSANTS

Résistances

Toutes des 5%, 50 V, 0,1 W, 0805
R1, R2, R3, R4, R5 = 10 kΩ
R6 = 470 Ω

Condensateurs

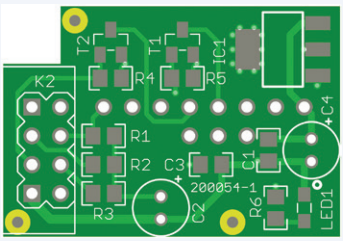
C1, C3 = 100 nF, 0805
C1, C4 = 10 µF, 16 V, pas de 2 mm

Semi-conducteurs

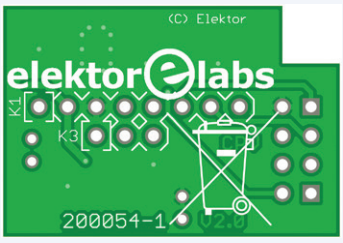
IC1 = LD1117AS33
LED1 = LED, verte
T1, T2 = BSS84

Divers

K1 = barrette mâle à 8 broches, pas de 2,54 mm
K2 = barrette femelle 2×4 broches, pas de 2,54 mm
K3 = barrette mâle à 3 broches, pas de 2,54 mm
Circuit imprimé 200054-1



Vue à 150% de la taille réelle



et fonctionne en 3,3 V. Ce module répandu, basé sur l'ESP8266EX est bon marché et facile à trouver (fig. 3).
Il y a cependant un mais. Les quatre ports GPIO de l'ESP-01 doivent être manipulés avec précaution, car ils déterminent également la manière dont le module fonctionnera à la mise sous tension. Pour un fonctionnement normal, les ports GPIO0, GPIO1 et GPIO2 doivent être à l'état haut à la mise sous tension, comme indiqué dans les notes dans le tableau « *Pin Definitions* » de la fiche technique de l'ESP8266EX. Beaucoup d'utilisateurs de l'ESP8266 le savent pour les GPIO0 et GPIO2, mais tout le monde ne le sait pas pour le GPIO1, alias TXD, la sortie série.

Association des ports GPIO aux signaux de commande

Grâce aux résistances R7 et R8, deux des quatre signaux de la carte de l'interrupteur, « Set » et « Reset », sont toujours à l'état bas à la mise sous tension. Lorsque la charge est alimentée au moment où la carte de l'interrupteur est mise sous tension, le signal « On » est aussi à l'état bas. N'oubliez pas que cela est possible parce que le relais est bistable et conserve son dernier état, même lorsqu'il n'est pas alimenté.
L'entrée S1 est flottante. Son état à la mise sous tension est déterminé par la carte de commande wifi, à moins que quelqu'un n'appuie sur le bouton-poussoir lors de la mise sous tension du système…
En résumé, nous avons donc affaire à quatre signaux qui peuvent tous être à l'état bas à la mise sous tension. Ils doivent être connectés à quatre ports, dont trois doivent être à l'état haut à la mise sous tension. Utiliser un module ESP-01 dans cette situation est donc un défi.

La solution : ajouter des transistors

Je suis parvenu à la solution suivante (**fig. 4**) :

- RXD (alias GPIO3) est la seule broche qui peut être connectée librement. Je l'ai donc connectée à la sortie de l'opto-coupleur car son niveau à la mise sous tension est imprévisible. RXD va être une entrée.
- Lorsque le GPIO0 est à l'état bas à la mise sous tension, le module démarre en mode de programmation Flash. Cela peut être utile pour mettre à jour le microprogramme, c'est pourquoi je l'ai connecté à S1. Par conséquent, GPIO0 va être une entrée aussi.
- Il reste alors GPIO2 et TXD (alias GPIO1) qui doivent donc devenir les sorties. Pour

les découpler des faibles impédances créées par R7 et R8 sur la carte de l'interrupteur, j'ai inséré des P-MOSFET avec une résistance de tirage sur leurs grilles. Cela garantit qu'à la mise sous tension, le GPIO2 et TXD verront un niveau haut. Après le démarrage de l'ESP-01, on peut les configurer en sorties à l'état bas.

Le pilote de sortie

Le pilote de sortie est illustré à la **figure 5**. Lorsque le signal GPIO passe à l'état bas, le P-MOSFET BSS84 conduit. Cela tire la grille du N-MOSFET BS170 vers le haut, donc il va conduire aussi et la bobine du relais est alimentée. Lorsque le signal GPIO est à l'état haut, le P-MOSFET BSS84 se bloque. Le N-MOSFET BS170 se bloque également à cause de la résistance de tirage de 1 kΩ sur sa grille, et la bobine du relais n'est pas alimentée. La résistance de 10 kΩ sur la grille du BSS84 assure que la broche GPIO est tirée vers le haut à la mise sous tension.

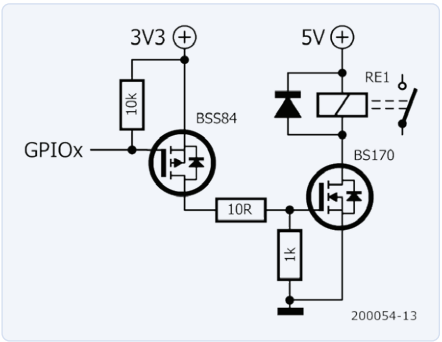


Figure 5. Le pilote de sortie : un P-MOSFET pilote un N-MOSFET.

Compléments

Pour l'alimentation en 3,3 V, j'ai simplement rajouté un régulateur à faible chute sur la ligne d'alimentation en 5 V (**fig. 6**). En pratique, R1, R6 et LED1 ne sont pas nécessaires car déjà présentes sur le module ESP-01, d'où la valeur « NC » (c.-à-d. non câblé). R5 n'est pas nécessaire non plus, mais je l'ai ajouté au cas où. La valeur appropriée pour R1 et R5 serait de 10 kΩ. R6 pourrait valoir quelque chose comme 470 Ω.
J'ai conçu un petit circuit imprimé pour l'interface entre le module wifi et la carte de l'interrupteur (**fig. 7**). Les fichiers se trouvent en [3].

Fichier YAML pour ESPHome

Une fois les entrées et les sorties définies, nous pouvons établir le fichier YAML de

Configuration YAML pour l'interrupteur wifi

```
esphome:
   name: wifiswitch
   platform: ESP8266
   board: esp01_1m

wifi:
   ssid: "my_ssid"
   password: "my_passphrase"

# Connexion impossible car les broches UART0
# sont utilisées pour les E/S
logger:
   baud_rate: 0 # UART désactivé.

# Activer l'API Home Assistant
api:

# Activer la programmation Over-the-Air
ota:

output:
   - platform: gpio
     id: relay_set
     pin: GPIO1 # TXD, peut être rappelé au niveau bas au démarrage.
     inverted: true
   - platform: gpio
     id: relay_reset
     pin: GPIO2 # Ne doit pas être rappelé au niveau bas au démarrage.
     inverted: true

switch:
   - platform: template
     name: "Wi-Fi switch"
     id: wifiswitch
     turn_on_action:
       # Impulsion sur la broche Set.
       - output.turn_on: relay_set
       - delay: 0.1s
       - output.turn_off: relay_set
     turn_off_action:
       # Impulsion sur la broche Reset.
       - output.turn_on: relay_reset
       - delay: 0.1s
       - output.turn_off: relay_reset
     # Utilise l'état de l'optocoupleur comme état de l'interrupteur.
     lambda: return id(optocoupler).state;

# Bouton-poussoir & optocoupleur.
binary_sensor:
   - platform: gpio
     name: "Wi-Fi switch pushbutton"
     id: pushbutton
     pin:
       number: GPIO0 # pas de rappel au démarrage.
       inverted: true
     on_press:
       then:
       - if:
           condition:
             binary_sensor.is_on: optocoupler
           then:
             switch.turn_off: wifiswitch
           else:
             switch.turn_on: wifiswitch
   - platform: gpio
     name: "Wi-Fi switch optocoupler"
     id: optocoupler
     pin:
       number: GPIO3 # RXD, peut être rappelé au niveau bas au démarrage.
       inverted: true
```

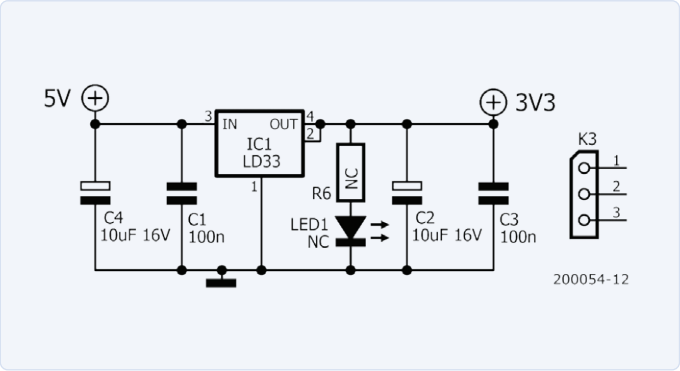


Figure 6. Un simple convertisseur de 5 V à 3,3 V. K3 est chargé de fournir un peu plus de stabilité mécanique lorsque le module wifi est branché sur le connecteur de la carte de l'interrupteur.

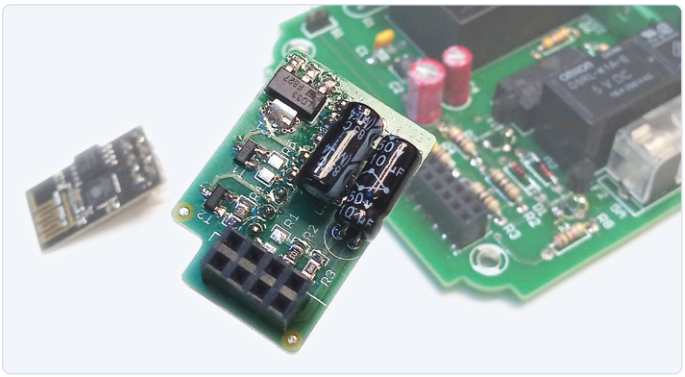


Figure 7. L'interface entre le module Wifi EPS-01 et la carte de l'interrupteur est disposée sur un petit circuit imprimé.

configuration d'ESPHome (voir l'encadré, disponible en [3]). C'est un peu plus compliqué que d'habitude en raison du relais bistable qui utilise deux broches au lieu d'une. ESPHome et Home Assistant connaissent un composant appelé « *cover* » qui peut commander de tels périphériques, mais il est destiné aux portes de garage, aux stores et autres qui ont un bouton d'ouverture et de fermeture. Ils ont également une icône spéciale. Notre interrupteur n'est qu'un interrupteur et nous voulons donc qu'il se comporte comme tel et qu'il en ait l'apparence. On peut le faire à l'aide d'un modèle d'interrupteur qui permet de spécifier séparément le comportement à la mise en marche et à l'arrêt.

Tout d'abord, nous définissons le GPIO1, alias TXD, comme une sortie inversée nommée *relay_set*. GPIO2 est déclaré comme une sortie inversée nommée *relay_reset*. Nous définissons également GPIO3 comme entrée numérique active à l'état bas pour la sortie de l'optocoupleur.

Ensuite, nous spécifions une courte impulsion de 100 ms sur la sortie *relay_set* comme action de mise en marche et nous faisons de même sur la sortie *relay_reset* comme action d'arrêt. Pour le retour d'état, nous renvoyons l'état de l'optocoupleur.

Pour que le bouton-poussoir S1 fonctionne comme prévu, c'est-à-dire qu'appuyer dessus commute la charge, nous définissons le GPIO0 comme entrée numérique active à l'état bas. Lorsqu'on appuie sur S1, on vérifie d'abord l'état de l'optocoupleur. Si la charge est active, alors on désactive le relais ; si la charge était inactive, alors on l'active. Une clause « *if-then-else* » fera l'affaire ici.

C'est fini !

Après avoir compilé ESPHome avec ce fichier YAML et flashé le module ESP-01, l'interrupteur wifi devrait apparaître dans Home Assistant, prêt pour l'automatisation. Il peut maintenant être installé et « commuter une charge ».

200054-04

Des questions, des commentaires ?
Vous avez des questions ou des commentaires sur cet article ? Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Idée, conception, texte et illustrations : **Clemens Valens**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

LIENS

[1] **L. Lemmens et M. Claussen, « nœud LoRa d'Elektor » , Elektor, mars/avril 2020** : www.elektormagazine.fr/180666-02
[2] **C. Valens, « la domotique, c'est facile avec… », Elektor, sept./oct. 2020** : www.elektormagazine.fr/200019-02
[3] **La page du projet** : www.elektormagazine.fr/labs/wifi-switch